A Simulation Tool for Virtual Laboratory Experiments in a WWW Environment

Alessandro Ferrero and Vincenzo Piuri

***Abstract*— Virtual instruments and distributed systems are of great interest to create advanced and flexible teaching and experimentation environments for measurement technologies at limited costs. Availability of simple and efficient technological supports to dissemination and remote use of virtual systems becomes attractive to increase the access to experimental practice without regard to the number of students and their location as well as the variety of instruments and measurement procedures directly available for experimentation.**

***Index Terms*—Distance learning, distributed laboratory, intellectual property, virtual instruments, world-wide web (WWW).**

I. Introduction

DURING recent years, the interest for didactic applications both of the software tools for the development of virtual instruments and the tools for the computer interconnection at great distance has increased steadily. This interest is mainly due to the cost of the experimental laboratories at universities with a large number of students. A second, but not less important reason, is the importance that teaching methods, based on the correct use of new technologies, are expected to assume in the near future. Besides, distance learning for continuous training is becoming a key factor to maintain the leading edge and to improve the quality of production, products and personnel in many small and medium sized enterprises.

From this point of view, instrument simulators are not actually expected to replace the real instruments. Real instruments and practical in-field experimentation cannot be replaced in the curricula of experimental subjects, like the electrical and electronic measurements and instrumentation. However, virtual instruments can be an extraordinary and powerful auxiliary didactic tool for the student, in order to help him become acquainted with the instrument and its controls and operations both in the class and remotely.

The actual effectiveness of these new didactic methods is, however, conditioned by the solution of the following practical problems.

- *Building simulators is a time-consuming job.* The software developing tools based on graphic, object-oriented programming methods, make this job easier, but are generally expensive. A run-time license must be purchased also for running the simulator, and not only the development tool. Careful measures must therefore be taken when the access to simulators is allowed from outside the university's campus.
- *The highest effectiveness of virtual didactic laboratories is attained when the laboratories can be accessed from outside the university's campus.* This way the students are given the possibility to revise the instruments' properties and characteristics at any time, and from any place (including from home).
- *The most adequate media for remote access is presently the Internet.* The simulators can be installed in a world-wide web (WWW) server [1], and the access to these resources can be organized as WWW pages. If this solution is adopted, the resident software must be adequately protected, and only a few files should be downloaded on the client computer, so that the access time to the server is also limited. These files should also be automatically removed from the client computer, at the end of the simulation session, so that the simulation software is protected, and no additional licenses are required.
- *Sharing of resources and experiences among universities.* The use of a global communication network, like the Internet, and high-level, highly standardized languages like HTML and Java allow for the allocation of the different simulation programs on different servers, possibly located at long distances from each other. This way, different universities with similar curricula might join in a developing team and distribute the burden of developing the simulators on a larger number of programmers. Each university might develop one instrument simulator, put it on its own WWW server, and make it available, through a proper link, to all other universities in the team.

This paper shows how the above ideas can be practically applied to realize a virtual laboratory, based on the well-known, widely-used environment LabView by National Instruments [2], and distributed through the Internet by means of the WWW services. Virtual instruments are realized in the standard way within the LabView environment, while distribution is assured both by a suitable WWW server [1] and software environment. An interface architecture is proposed between LabView and WWW environments, as well as an access method to the WWW server, especially developed in order to guarantee a high level of protection of the resident software and minimize the server access time.

II. Specifications of the System Functionalities

A virtual laboratory for measurement and instrumentation must aim to realize an integrated environment for creation and distribution of simulators of instruments and systems through a

Manuscript received June 11, 1998; revised February 26, 1999.

The authors are with the Dipartimento di Elettrotecnica, Politecnico di Milano, 20133 Milano, Italy.

Publisher Item Identifier S 0018-9456(99)04987-6.

0018–9456/99$10.00 © 1999 IEEE

computer network. Availability of component libraries allows the user to dynamically build his/her own measurement bench by using a personal computer only.

The main goals pursued in our research and experimental implementation were

- definition of a modular approach to virtual instrument and bench design;
- definition of a standard environment and tools to create the instrument and bench components;
- definition of a standard interface among component simulators to allow easy interfacing, even if realized by different persons;
- creation of an instrument and component database;
- definition of a distribution system supporting sharing of components and simulators;
- definition of a distribution environment complying with the most-used computer network standards;
- definition of a software environment supporting workgroup and remote didactic activities;
- realization of a user-friendly interface;
- definition of a flexible approach to experimentation;
- realization of a simulation engine allowing dynamic connection of components;
- support of contemporaneous access by many users;
- identification, control and monitoring of the users and all accesses to the system;
- efficiency of the simulation by adopting a decentralized strategy;
- efficiency of the network connection;
- protection of the copyright of the simulation software;
- protection of the distribution server from possible external attacks;
- identification of a low-cost hardware and software architecture for wide and inexpensive use of the distribution environment and simulators.

III. System Architecture

To achieve the goals mentioned in the previous section, we designed and implemented a client–server distributed environment [3], as shown in Fig. 1. The server and the clients are connected on the same local area network (LAN) within the laboratory or within the same campus. Remote connections may be set up between the server and a remote laboratory as well as a single remote user working at home; in these cases, networking must be realized to reproduce exactly the same kind of interconnection available within a LAN.

The WWW technology is used to realize the hypertext distribution system for the simulator components and engine. The client downloads the desired components and the simulation engine from the WWW server. Components are assembled on the client to create a single instrument or a whole bench. Components are created by using LabView by National Instruments and stored in the server database. Simulation is performed directly on the client. Security is provided by user authentication on the server. Software protection is achieved by continuous checking of the connection between clients and server.

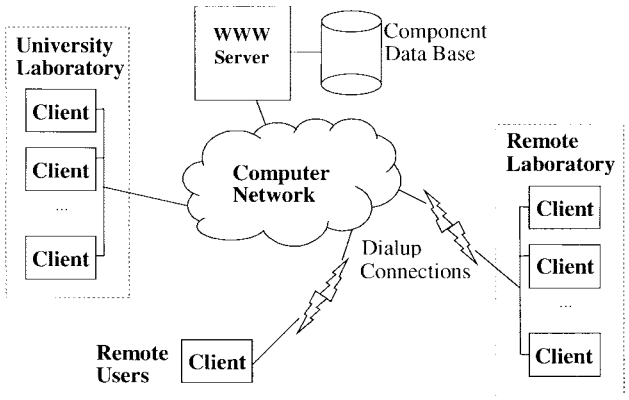

Fig. 1. Client–server hardware environment.

To achieve a standard component distribution system, we adopted the WWW technologies, including the HTML, CGI, and Java languages, allowing portability and independence through different client hardware/software architectures. Availability of standard portable languages is in fact instrumental to provide independence of the application from the client system on which it is executed. Since the languages mentioned above are interpreted and not compiled, instructions are not definitely bound to a given hardware architecture and to a given operating system. They can be moved through the computer network without any problem, needing only the presence of a software interpreter (the Internet browser suitable for the WWW technologies) on the receiving computer: only this interpreter must be compiled for the specific hardware and operating system. As a consequence, it is necessary to realize only one version of each instrumentation component. The component will then be able to run on any computer architecture (from a personal computer to a workstation and even to a larger computer with users connected through X-window terminals) as well as on any operating system (from MS-DOS with Microsoft Windows, to Microsoft Windows 95, to Microsoft Windows/NT, to the different versions of Unix). The WWW browser can now be considered a standard component of any computer installation: therefore, our approach will automatically fit on any hardware/software environment widely available both in the university laboratories and at home for remote training.

The multimedia technology supported by the WWW browsers allow creation of more sophisticated user interfaces, by simulating the real instruments and components in a realistic way.

The software structure of our environment, both on the server and on the client, is shown in Fig. 2. In our experimental realization, the server is realized by using a Hewlett-Packard 9000 715/50 workstation running HP-UX 9.0 with TCP/IP network protocol, the mSQL database by Hughes Technologies, and the HTTPD WWW server by NCSA with some configuration extensions [4]. The clients are IBM-compatible personal computers running MS-DOS with Microsoft Windows 3.11 as well as MS-Windows 95; the WWW browsers are Netscape Navigator 3.0 and Microsoft Internet Explorer 3.0, or higher versions. The virtual laboratory environment on the server

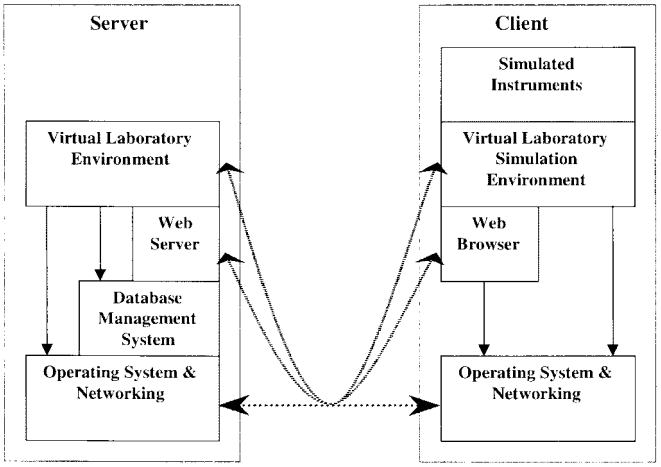

Fig. 2. Client–server software environment.

manages the distribution of the simulation engine and the instrument components. The virtual laboratory environment on the client is the simulation engine executing the virtual instruments created by the user.

IV. Building Virtual Instruments

To guarantee future diffusion and enhancement of our distributed virtual laboratory, we prefer de facto standards to implement the simulated instruments. In particular we adopted LabView by National Instruments since it is the most widely-used software environment to create virtual instruments both in schools and industry.

Our goal is to create virtual benches of instruments and signal generators on which students can practically observe and try their procedures, their characteristics, and their use. The basic structure of the measurement workbench is composed therefore of the following components:

- virtual signal generators, simulating the real systems of which we like to measure physical quantities;
- virtual instruments, emulating the measurement operation performed by real instruments on the physical quantities.

In the virtual environment provided by LabView, the workbench can be easily realized on workstations or personal computers. First of all, the definition of each component must be created by using the visual editor, which is available in LabView: inputs and outputs are identified, and the operations to be performed on inputs to generate the outputs are specified. To share definition of components, it is necessary to store them on a file server so that all users can easily download the desired ones. This also simplifies distribution of the components within the whole laboratory through the use of computer network supports. When the student wants to experiment with the virtual workbench, he/she must select the desired components and must connect them together within the LabView environment, by linking the inputs of each component to the outputs generated by other components. This resembles the operations physically performed in setting up a real workbench by using real generators and instruments. Finally, the workbench can be simulated, by starting the simulation engine embedded in LabView. Local simulation on the individual workstation should usually be preferred since it allows for higher performance, higher interaction, higher flexibility, and lower hardware and software costs.

The above operations are executed easily by students with non marginal skills on LabView, computers, and networking. However, they may become insuperable obstacles for other students so that the effectiveness and wide usability of practical experimentation and continuous training may be limited drastically. Besides, linking the components on the workstation to generate the executable code of the workbench calls for the availability of a development license of the LabView environment for each workstation, due to software copyright rules. On the other hand, linking the components on the server generates an executable code of the workbench that must incorporate the run-time support of the LabView environment to be executed on the workstation without local installation of the LabView development system. Software copyright rules impose acquisition of a sufficient number of run-time licenses, but this number may increase dramatically since distribution, subsequent dissemination and use of the executable code of the workbenches cannot be controlled. Moreover, the executable code of the workbenches could even be sold by people creating them in the university laboratory without paying the royalties for the embedded run-time support to National Instruments and without paying any fee for the use of the component database developed by the university.

Therefore a guided creation of the workbench becomes mandatory as well as the control of the distribution of the workbench itself to protect the intellectual property of the components, the simulation environment, and the LabView engine. To achieve the above goals, we split creation of the workbench and the related executable code into separate phases, which are executed partly on the WWW server and partly on the client in cooperation with the server.

The component definitions are created off-line by experts in the instrumentation and measurement procedures, by using the visual editor of LabView. These definitions (which are not yet executable) are stored in files in the component database of the WWW server. They are in a format that can be read and interpreted by LabView as any other interpreted language, instead of being compiled with the run-time support and executed by the workstation processor.

The student will simply need to select among such predefined components representing instruments or generators to build his/her own workbench. Selection is supported by an easy-to-use WWW interface based on graphics and visual operations. To simplify dynamic linking of the selected components, a standard bench framework is provided. It is a virtual instrument definition, which is written in the interpreted language of LabView and contains the space for two signal generators and two instruments (this setup is typical of most of the laboratory experiments in the area of electrical and electronic measurement). The user must only select which specific component to place in each of these slots: the corresponding definition file is associated to the slot as an external procedure to be interpreted during simulation. The system manager can easily modify the component number by

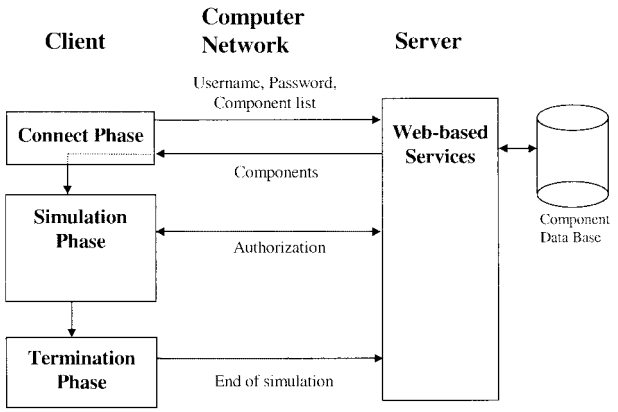

Fig. 3. Operation of the simulation environment.

changing the characteristic parameters of the standard bench framework available on the server; users must reload the framework so that changes take effect.

To perform simulation of the components in the virtual workbench, a simulation engine must be used with the selected component definitions. Our engine contains the run-time support of LabView configured to interpret the standard bench framework with the component definition selected by the user: such definitions are viewed as an external procedure dynamically linked to the framework and interpreted by the engine itself. The engine also contains suitable checking points in order to verify the user authorizations. Interpretation is slower than direct execution of executable code, but guarantees modularity and protection of the intellectual property.

V. Networking Creation and Simulation of Virtual Instruments

To download the simulation engine and the instrument components as well as to create the virtual instruments and to guarantee the intellectual property, modular interconnection and management of clients and server are necessary.

The user activates and operates in the virtual laboratory by performing the following steps, as shown in Fig. 3:

- connect to the home page of the WWW server;
- identify the user authorization through username and password and obtain the use of a floating license of the virtual simulation software;
- select the instrument components available in the database from the instrument menu;
- dynamically link the selected instrument components to create the desired virtual instrument to be simulated on the client;
- download the created virtual instrument,
- activate the simulation engine on the client with the created virtual instrument;
- terminate the simulation session by releasing the server connection and the floating license.

The first time the client WWW browser connects to the WWW server, the simulation engine (about 1.2 megabytes) is automatically downloaded from the server itself. This avoids the need of distributing the software for the virtual laboratory through floppy disks or manually from the server. The self-decompressing and self-installing engine creates the necessary directories, executable files and configuration data on the client. The WWW browser configuration is automatically updated to allow direct recognition and management of the virtual instrument bench created on the server. This is obtained by introducing the standard file extension `.VIS` identifying such files. During subsequent environment activation, the WWW browser on the client will be able to treat these files automatically. Whenever one of them is clicked by the user, it is downloaded from the server; then, the management program extracts and decompresses the components, sets up the simulation environment, and starts the simulation engine by feeding it with the virtual instrument bench definition without any further user operation.

When the WWW browser connects to the WWW server, a welcome page asks the user to provide his/her username and password for authentication. These data are sent to a CGI script on the server for analysis and database comparison.

The user can now select the desired instrument components among the ones currently available in the component database. Since the whole workbench is virtually simulated, i.e., not only the instruments but also the measured signals, the user must also select the desired simulators of virtual signal generators. These components are connected in the standard framework provided by the system. Configuration of the virtual bench is now sent to a CGI script on the server for analysis, dynamic linking of the selected components, and creation of the file containing the virtual bench definition for the simulation engine. The file is stored in a compressed format to limit the network overhead during downloading. The server holds the workbench files of each user in a separate directory to allow subsequent reuse of definitions while avoiding overriding among different users. An HTML page is generated and presented to the user: it contains the link to the files on the server for the simulation engine. By clicking on such a link, the user can download such a file (about 100 kb) and start the simulation as discussed above.

The simulation engine executes the virtual instrument definition created by the user. At first, the engine reads the configuration file and sets up its internal parameters and configuration to accommodate the instrument structure for the simulation. Then, it loads the definition of the instrument components and creates the data structures supporting efficient simulation. The instrument simulation is performed and the connection to the server is verified for periodic authorization checking. During the first part of each simulation cycle, the signal generators are executed to create input signals for the instruments, which are run during the second part of the cycle. The simulation continues until the user hits the OFF or the EXIT buttons. In the first case, the simulation is suspended; in the second one, it is terminated, the whole environment is cleared, and the server connection closed.

To support the interconnection of clients to the server, we used standard protocols widely adopted in the worldwide community. This allows any perspective user of the simulation environment for virtual laboratories to simply plug his/her client computer to the international network and to obtain the

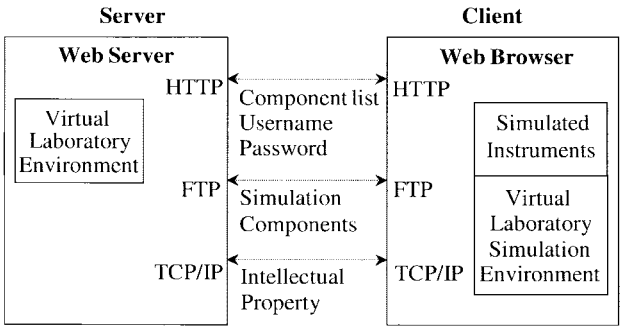

Fig. 4. Network protocol hierarchy.

simulation environment and the instrumentation components directly from the server without any preliminary acquisition of specific software.

The standard protocols for networking adopted in our work are at different levels as shown in Fig. 4 [1], [3], [5].

- TCP/IP defines the physical interconnection, data transfer and message routing management. This is the typical protocol suite adopted in any Unix network as well as in the standard Internet.
- FTP specifies the file transfer method between computers on the TCP/IP protocol suite. It is used to move long messages containing files around the network: in our case, it is used to transfer the simulation engine and the components from the server database to the clients.
- HTTP is the application protocol defining the connection to the web server and the transfer methods for hypertexts between client and server by using the TCP/IP protocol suite as well as the FTP whenever necessary for large files of data.

One of the most critical issues in using a networked environment over a standard open network is security [3], [5], [6]. This widely studied problem concerns the privacy of data and programs (in our case, the necessity of guaranteeing the intellectual property both of the simulation engine and the instrument components as well as the passwords), the dissemination of viruses, and the attacks of hackers.

To guarantee privacy, encryption methods can be adopted to protect the communications between server and client; this requires that both the server and the clients use the same secure data encryption mechanism. In our system, we used the secure HTTP (SHTTP) protocol, which is integrated in the standard HTTP environments, even if it is specific for hypertexts. Higher security techniques could be envisioned, but their costs and the worldwide availability limit their applicability. To protect the copyright on the simulation engine software, we adopted the concept of floating license. A given number of licenses are associated to the server and paid by the server owner. When a connection is authorized, one of the licenses is considered used by such a connection: no more than the given number of licenses can be contemporaneously running, i.e., no more than such a number of users can use the simulation environment of the virtual laboratory. To guarantee that this user limit is never exceeded, the engine running on the client frequently contacts the authorization daemon on the server to verify the user authorization: if negative or no answer is received from the server, the simulation engine aborts its operations. Authorization is validated by encrypted message exchange between the client and the server, according to a suitable encoding and checking protocol. This reduces the overall data transferred between server and client at the beginning of the simulation, because the engine can be freely distributed and stored on the clients without breaking copyrights. Since it is not able to run without the controlled server connection, only the predefined number of clients can in fact operate at any time.

To protect the server, several techniques can be taken into account, e.g., access restriction based on user identification, firewalls, and encryption. For simplicity and cost reasons with respect to their effectiveness and the relevance of the system to be protected, we adopted an approach based on access restriction through user validation: both the password and the IP address have been included to control the user operations. Obviously this implies some management overhead since it becomes necessary both to grant passwords and register enabled IP addresses. A suitable protected HTML page is provided on the server to simplify the authorization management. Unfortunately, checking the IP addresses forbids connection of a perspective user through a dialup line by using SLIP or PPP protocols with automatic dynamic assignment of IP addresses. Accurate software engineering techniques and test procedures are then required to write good and reliable CGI scripts for the client-server application connection, in order to avoid bugs allowing undesired entrance to possible hackers.

VI. Conclusions

The paper presented a simulation environment supporting experiments in a virtual laboratory by using WWW technologies. The use of de facto standards and user-friendly solutions allows a high number of people (even with minimum knowledge on computers) to access the system and benefit from advanced training without requiring creation of huge and expensive laboratories with many working benches. Only few real instruments will be required in the laboratories to provide the final refinement of practice and operating abilities of the students. In turn, this allows universities and other educational environments to reduce the investments for each kind of equipment and, as a consequence, to increase the number of innovative and different types of equipment, with obvious benefits for the students. The system was tested successfully in a virtual laboratory with 20 clients operating contemporaneously as well as for remote dialup connections at 14 400 bps.

Further developments are directly supported by our WWW standard approach. The hypertext technologies of the WWW solution allow for creating navigation paths through the material available on the web site. For example, explanatory and help pages can be attached to the components in order to provide enhanced on-line tutoring to students, to answer indirectly most of their questions on the real instruments and on the simulation environment arising during the use of the simulator and the virtual systems.

Besides, a hypertext book can be realized to guide and support the students with advanced self-training approaches, a complementary tool for traditional teaching. Within such a book, experiments can be set up to test immediately the acquired knowledge and verify the comprehension degree in the field. The multimedia technologies are, in this case, instrumental to enhance the readability and understandability of the book by reconstructing as much as possible the real working environment through images, graphics, short movies, sounds, and so on.

Such learning approaches are becoming more and more attractive in our society where continuous training is the key factor to maintain the leading edge in any industrial area due to the evolving technologies as well as to qualify the personnel for new and alternative tasks. Availability of systems supporting practical experience at low cost through simulation on any standard low-cost computing architecture (e.g., the personal computers) is relevant to guarantee extensive use and facilitate remote training, even beyond the working hours and traditional educational environments.

A further extension concerns distributed creation and maintenance of the component database. Due to the intrinsically distributed nature of the environment, several educational institutions can cooperate to realize a higher number and more accurate virtual components that can be shared through the network, by exploiting the specific abilities and knowledge of each institution.

Finally, the modular structure of the virtual laboratory environment allows for introducing more realistic experiments by replacing the simulated signal generators with acquisition boards to measure quantities in the field. On the other hand, intrinsic distribution also allows for introducing remote acquisition boards distributed over the network for remote measurement.

ACKNOWLEDGMENT

The authors would like to thank N. Simeoni and P. Teruzzi for their support in the experimental phase.

REFERENCES

[1] J. Niederst, *Designing for the Web*, O'Reilly & Associates, Inc., 1996.
[2] *LabVIEW User Manual*, National Instruments, 1996.
[3] [3] A. S. Tanenbaum, *Computer Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[4] NCSA HTTPd Development Team, "NCSA HTTPd tutorials," available at http://hoohoo.ncsa.uiuc.edu/docs/tutorials/, 1996.
[5] D. Comer, *Internetworking with TCP/IP*. Englewood Cliffs, NJ: Prentice-Hall, 1994.
[6] V. Ahuja, *Network and Internet Security*. Boston, MA: Academic, 1996.

Alessandro Ferrero received the M.S. degree in electrical engineering from Politecnico di Milano, Milano, Italy, in 1978.

In 1983, he joined the Dipartimento di Elettrotecnica, Politecnico di Milano, as Assistant Professor of Electrical Measurements. From 1987 to 1991, he was Associate Professor of Measurements on Electrical Machines and Plants, University of Catania, Catania, Italy. From 1991 to 1994, he was Associate Professor of Electrical Measurements, Dipartimento di Elettrotecnica, Politecnico di Milano, where he is presently Full Professor of Electrical and Electronic Measurements. His current research interests include application of digital methods to electrical measurements and measurements on electric power systems.

Mr. Ferrero is a member of the Italian Informal C.N.R. Group on Electrical and Electronic Measurements, and Member of the Administrative Committee of the IEEE Instrumentation and Measurement Society.

Vincenzo Piuri received the Ph.D. degree in computer engineering in 1989 from the Politecnico di Milano, Milano, Italy.

Since 1992, he has been Associate Professor of Operating Systems at the Politecnico di Milano. His research interests include distributed and parallel computing systems, theory, and applications of neural techniques. Results were published in more than 120 papers in book chapters, international journals, and proceedings of international conferences.

Dr. Piuri is Founding Co-Chair of the Technical Committee on Emerging Technologies and the Technical Committee on Intelligent Measurement Systems of the IEEE Instrumentation and Measurement Society. He is an Associate Editor of this TRANSACTIONS. He served as Program Co-Chair for IMTC'99.